6 NMRi I/O

6.1 Overview

The NMRi I/O routines are provided to allow the transfer of data between GAMMA and the programs NMR1 and NMR2[1].

6.2 Available NMRi Functions

NMRi - Write an NMRi file page 106
NMRi_1D - Read an NMRi file into a GAMMA vector page 108
NMRi_2D - Read an NMRi file into a GAMMA matrix page 108
NMRi_header - Read the header of an NMRi file page 109

6.3 Routines

6.3.1 NMRi

Usage:

```
#include <NMRi.h>
void NMRi (const char* filename, block_1D &BLK, int rc=1);
void NMRi (const char* filename, block_2D &BLK, int rc=1);
void NMRi (File &file, block_1D &BLK, int rc=1, int zero=0);
```

Description:

The function NMRi is used to write files in the standard NMRi format. The data is contained in the data block “BLK”. The file is specified by either the name “filename” or by the file pointer “file”. The parameter “rc” indicates whether to write real or complex(default) data. If rc is 0, only the real data will be output. If rc is negative, only the imaginary data is written (as reals), and if rc>0 the complex data is written.

1. NMRi (const char* filename, block_1D &BLK, int rc=1) - When NMRi is invoked with this argument list it writes the data contained in the 1-dimensional data block “BLK” to a newly constructed file “filename” in the NMRi format. If rc is 1 (default) the data is written as complex numbers. If rc is set to zero, only the real data of the block is output. Any parameters in BLK compatible with the NMRi parameters will be automatically transferred into the NMRi file. The file is closed upon the function return. The func-

1. NMR1 and NMR2 are a products of New Methods Research, Inc., 7 East Genesee Street, Syracuse, NY, 13210. Phone: (315) 424-0329, FAX: (315) 424-0356.GAMMA was tested on Sun systems running the Sunviews operating system. NMR1 release 3.8 and NMR2 release 3.5 were the manuals referred to during the programming.

tion will overwrite any file “filename” previously in existence. No other NMRi file should be open during the calling of this function.

2. NMRi (const char* filename, block_2D &BLK, int rc=1) - This function is similar to the use above (with 1D data blocks) but writes the data contained in the 2-dimensional data block “BLK”. This is done row-wise, i.e. first row 1 of BLK is written followed by row 2 and so on until the end of the data in BLK.
3. NMRi (File &file, block_1D &BLK, int rc=1) - When NMRi is invoked with this argument list it writes the data contained in the 1-dimensional data block “BLK” in the NMRi format to wherever the file pointer “file” is at. The file where “file” is pointing is assumed to be open (opened explicitly sometime prior to the function call) and will remain open with the file pointer advanced to the file end at the function return. The file should be closed externally as well. The NMRi parameter block is updated with each function call. An optional parameter in this function is “zero” which, if non-zero, will set the count of 1D spectra present in the NMRi file to zero. This is done automatically in most instances but should be used if multiple NMRi files are to be produced in a single program run it is necessary to re-zero the count when writing the first spectrum of all NMRi files.

Note that NMRi can be called with a matrix instead of a block_2D or a row_vector rather than a block_1D. In these instances the parameter sets are not fully written into the NMRi file.

Return Value:

Nothing when producing a NMRi file.

Example:

```
#include <gamma.hc>
main ()
{
 block_1D BLK1(128);                  // Define a 1D-data block
BLK1 = Gaussian(128, 64, 42.46);      // Set 1D-block to Gaussian
NMRi(“NMR1D.tst”, BLK1);              // Output 1D BLK to NMRi file
block_2D BLK2(128,128);               // Define a 2D-data block
for(int i=0; i<128; i++)              // Fill 2D block with 2D-Gaussian
  for(int j=0; j<128; j++)
    BLK2(i,j) = BLK1(i)*BLK1(j);
NMRi(“NMR2D.tst”, BLK2);              // Output 2D BLK to NMRi file
File fp;                              // Specify a file pointer
fp.open(“NMRND.tst”,io_writeonly
                              ,a_create); // Open file NMRND.tst
block_1D BLKA(128), BLKB(128);        // Define two more 1D-data blocks
BLKA = sinc(128, 64, 12);             // Set block to sinc function
BLKB = 127-i * BLK1;
NMRi(fp, BLKB, 0, 1);                 // Last 1 needed as previous NMRi files used
for(i=1; i<128; i++)                  // Output 128 1D blocks to NMRi file
  {                                   // only the reals are output here
  BLKB = i*BLKA + (127-i)*BLK1;
```

```
  NMRi(fp, BLKB, 0);
  }
fp.close();                              // Close file NMRND.tst
}
```

The last NMRi file produced, NMRND.tst, was made by first opening a file, successively writing 1D spectra to it, then closing the file. Since this program accessed NMRi files previously (during the creation of NMR1D.tst and NMR2D.tst) the first spectrum was independently written outside the loop with the zero flag set when calling NMRi. This sets the internal counter to zero, something unnecessary had NMRi files not been previously created in the program.

6.3.2 NMRi_1D

Usage:

```
#include <NMRi.h>
block_1D NMRi_1D (const char* filename, int spectrum=1);
```

Description:

The function NMRi_1D is used to read a 1D spectrum from an NMRi file, that is, a file in the standard NMRi format. The NMRi file name is “filename”. Returned is an 1D-data block containing the parameters in the NMRi file along with the spectrum chosen with “spectrum”. The default spectrum will be the initial one.

Return Value:

A one dimensional data block.

Example:

6.3.3 NMRi_2D

Usage:

```
#include <NMRi.h>
block_2D NMRi_2D (const char* filename);
```

Description:

The function NMRi_2D is used to read a 2D spectrum from an NMRi file, that is, a file in the standard NMRi format. The NMRi file name is “filename”. Returned is an 2D-data block containing the parameters in the NMRi file along with the entire 2D spectrum.

Return Value:

A two dimensional data block.

Example:

Copyright Scott A. Smith

6.3.4 NMRi_header

Usage:

```
#include <NMRi.h>
ostream& NMRi_header (const char* filename, int verbose=0);
```

Description:

The function NMRi_header is used to quickly read the header information contained in an NMRi file. The NMRi file name is “filename”. The header contains 512 pieces of information, only some of which is particularly useful outside of the NMRi programs themselves. The integer verbose specifies how much of this information should be returned. The default value of 0 returns the minimal amount of information, the amount increases as verbose increases up to 10. The header is returned in an output stream.

Return Value:

None, the function sends information to standard output

Example:

```
#include <gamma.h>

main ()
{
String filename;                          // Declare a string
cout << “Which Levy File ?”;              // Ask for filename
cin >> filename;                          // Input filename
int verbose;                              // Declear an integer
cout << “\nHow verbose ?”;                // Ask for verbose level
cin >> verbose;                           // Input verbose level
cout << “\n\n”;                           // Output line feed (so screen is nice)
NMRi_header(filename, verbose);           // Output the header info in filename
}
```

In this example, the user is queried for the file and the amount of information should be returned. The default value of 0 returns the minimal amount of information, the amount increases as verbose increases up to 10. The header information is returned to the standard output stream.

Copyright Scott A. Smith

6.4 Description

NMRi File Structure - NMRi files begin with a header which contains 512 parameters describing the spectral data. A 1D-spectrum of length “SIZE” is written immediately after the header, all real values followed by all imaginary values. If the data is not complex, the imaginary values are absent from the file. A two (or multi-) dimensional spectrum is successive 1D-spectra sharing the same header information, as depicted in the figure below.

NMRi File Structure

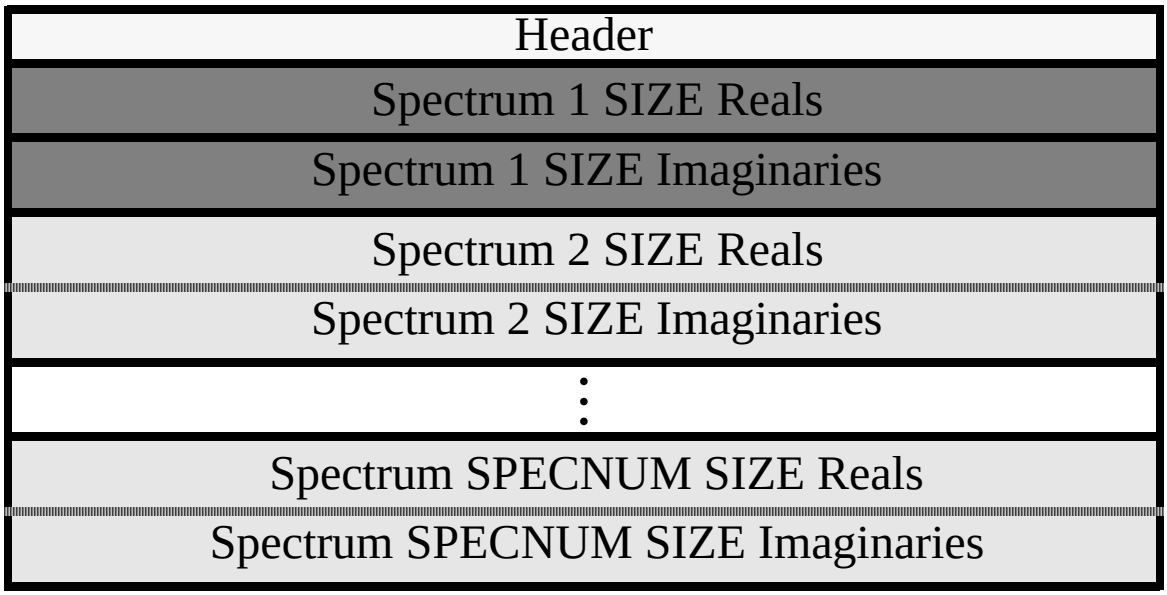

Figure 25-1 - Overall file structure of a NMRi file containing SPECNUM 1D complex spectra each of length SIZE.

This file structure is described in the NMRi manual *LAB ONE NMR2*, USER MANUAL, Release 3.5 on page 29.

NMRi Header Structure - A full listing of the parameters contained in the header is given in the NMRi manual *LAB ONE NMR1*, USER MANUAL, Release 3.8 on page 1-37. Of the 512 parameters it contains, few are useful outside of the NMRi programs. Those absolutely required are given in the NMR2 manual (cited previously) on page 28 and listed here.

- 100 - First dimension size. This is SIZE in Figure 25-1
- 101 - First dimension spectral width.
- 107 - Data type: 0.0=quadrature, 1.0=singulature, 2.0=singulature sorted.
- 220 - Second dimension size. This is SPECNUM in Figure 25-1
- 230 - Second dimension spectral width.

For consistency with the program description found in the supplied manuals, the file structure will be briefly discussed in terms of blocks, each block being of length 64[1]. Thus the header takes up 4 blocks.

NMRi Header Structure

Figure 25-2 - Header Structure of a NMRi file. The header is conceptually divided into 4 blocks, each containing 64 parameters.

NMRi Data Structure - The actual data starts immediately after the header. In a file containing multiple spectra (i.e. a 1D spectrum) each spectrum must be of the same size and this size must be a power of 2 (64, 128, 256, ...). If the any spectrum does not meet this latter requirement it will be padded with zeros until the size is a power of 2. Furthermore, they must all be real or all be complex.

NMRi Data Structure

Figure 25-3 - Structure of NMRi 1D-spectrum.

Again, for consistency with the program description found in the supplied manuals, the file structure will be discussed in terms of blocks of length 64[2]. The data begins at block number 4, and the length of each 1D-spectrum will be

$$LENGTH = (SIZE)/64$$

for a complex data set where LENGTH is given in blocks. For a real data set, this number will be halved.

GAMMA - When creating NMRi files, GAMMA will initially write an empty header (512) prior to any spectral data. After each 1D-spectrum is written, the header will be updated to reflect how many lines of 1D data are present. That would be the variable SPECNUM shown in Figure 25-1.

1. The block length of 64 is intrinsic to the file structure as utilized in the FORTRAN code for file I/O supplied with the NMRi programs.GAMMA uses C++ code to perform I/O withe NMRi making this conceptual division nearly unnecessary.
2. See the previous footnote. This information is only useful in clarification of the manual description.

Copyright Scott A. Smith